Python

Лекция 1

Преподаватель: *Дмитрий Косицин*

О курсе

Формат курса

- 10 лекций + 10 практических занятий [+ 7 семинаров]
- Лабы + 2 или 3 домашки (уже скоро!)

Что ожидается

- Все материалы доступны в anytask
- См. учебную программу
- Будет много советов =)

e-mail для связи: u@trix.by

О языке

...

История

Автор: Гвидо ван Россум

Язык появился в 1991 году.

- 1994 год – Python 1.0
- 2000 год – Python 2.0
- 2008 год – Python 3.0

Особенности

- Free, OpenSource, Portable (cross platform)
- Неплохая стандартная библиотека

- Высокоуровневый
- Интерпретируемый
- Объектно-ориентированный
- Жесткая динамическая типизация

Эффективность

- Понятность кода
  - Легко писать
  - Легко читать
  - Легко отлаживать
  - Есть официальный style guide (PEP-8, PEP-257)
- Низкая эффективность
  - Чистый Python медленнее C++ в 5-100 раз
- Использование
  - Прототипы
  - Интеграция / вспомогательные программы
  - Исследования

Python Zen

```
import this
```

- Beautiful is better than ugly.
- ...
- Simple is better than complex.
- Complex is better than complicated.
- ...
- Readability counts.
- Special cases aren't special enough to break the rules.
- ...

Версии и интерпретаторы

Версии

- Существует 2 официальные несовместимые версии Python: 2.x и 3.x
- 3.x активно развивается (3.7.0a – последняя)
- 2.x поддерживается (2.7.14rc1; some features are backported from Py3)
- 2.x используется во множестве компаний

Интерпретаторы

- CPython
- IronPython, Jython
- PyPy

Среды разработки

- SublimeText + python
- IPython [Notebook] / Jupyter
- PyCharm Community Edition
- Visual Studio with plugin (native in VS 2017)

Настройка и установка

Linux/MacOS

- Обычно предустановлен
- Разные версии в менеджерах пакетов
- Самая свежая ⟶ установщик/исходники с сайта

Windows

- Установщик с сайта

(можно держать несколько версий в системе)

Системная переменная путей

`PATH` – набор путей, по которым происходит поиск запускаемой программы (если не указан абсолютный путь)

Просмотр содержимого: **`echo $PATH`** (Linux/MacOS) or **`PATH`** (Windows)

Выбранный Python: **`which python`** (Linux/MacOS) or **`where python`** (Windows)

Проблемы:

- Несколько разных версий Python
- Переменная **`PYTHONPATH`**

Установка библиотек

Индекс пакетов PyPI (Python Package Index):

- Есть все популярные библиотеки
- Установка с помощью **`pip`** (out from the box для последних версий)
- Полезно заглянуть на сайты библиотек (документация / инструкции)

Возможные проблемы:

- Компиляция библиотек
- Неправильные пути в **`PATH`** и/или **`PYTHONPATH`**

Используйте **`virtualenv`** для поддержки различных версий пакетов.

Дистрибутивы: Python(x,y), Anaconda, Canopy

Основы языка

Интерпретатор

Обычный режим

Интерпретатор + исходный код (новый процесс с программой)

```
$ python main.py
```

Интерактивный режим

Интерпретатор без кода (исполнение кода online)

```
$ python
```

`help(...)` – справка по указанному объекту (выход: **`q`**)

`exit()` or **`quit()`** – выход

Как это работает?

```
>>> print 'Hi, guys!'
Hi, guys!

>>> for i in xrange(1, 7):
>>>     if not i % 3:
>>>         print i ** 3
27
216
```

Python как калькулятор

- Типы: **int (long), float, complex**; *дополнительно* ***Fraction*** *и* ***Decimal***
- Арифметические операции:

+ – * ** / % *(divmod)*

```
>>> -2 * 7 / 3 + 4 ** (4 % 2)
-4
```

- Особенности деления:

```
>>> print 7 / 3, 7 / 3.0, 7 // 3.0
2 2.33333333333 2.0
```

Отличия Python 3

- Особенности деления и **print** (их можно подключить в Python 2.x):

```
>>> print(7 / 3, 7 / 3.0, 7 // 3.0)
2.33333333333 2.33333333333 2.0
```

- Оператор матричного умножения @ (PEP465, Python 3.5+)

```
>>> x = numpy.array([ 1., 1., 1.])
>>> m = numpy.array([[ 1., 0., 0.],
                     [ 0., 1., 0.],
                     [ 0., 0., 1.]])

>>> x @ m
numpy.array([ 1., 1., 1.])
```

Замечания о числах

- Битовые операторы:

<< & | ^ >>

- Для задания чисел в других системах счисления используются префиксы:

```
>>> 0b1001 == (1 << 3) + 1
True
```

Логические выражения

```
>>> 0 != 0
False
>>> 2 * 2 == 4
True
>>> False or 0 > -1 and True
True
>>> not 0 < 1 <= 5
False
```

Переменные

```
>>> x = 1
>>> y = 2.5
>>> x + y
3.5
>>> True and y
2.5
>>> y /= 2 * x   # y = 1.25 (yep, it's a comment)
```

Условный оператор if

```
>>> if a < 0: # first way
>>>     b = -a
>>> elif a > 0:
>>>     b = a
>>> else:
>>>     b = 0

>>> b = a if a >= 0 else -a   # second way

>>> b = abs(a)   # third way
```

Цикл while

```
>>> attempt_count = 0
>>> max_attempt_count = 5
>>> while attempt_count < max_attempt_count:
>>>     # convert from str to int
>>>     x = int(raw_input('enter an integer: '))
>>>     if x > 20:
>>>         break
>>>     attempt_count += 1
>>> else:  # if cycle hasn't been broken
>>>     print ('expected integer has not been received'
>>>            ' after %d attempts' % max_attempt_count)
```

Цикл for

```
>>> # no need to maintain counter
>>> for i in xrange(1, 5, 2):
>>>     print i, i + 1,
1 2 3 4
```

- **`range(5)`** vs **`xrange(5)`** – список / итератор (разберем позднее)
- **Важно!** Нельзя изменять ***итерируемый объект*** (здесь – **`xrange`**) в теле цикла!

Функции

```
>>> def print_hello():
>>>     print 'hello'

>>> print_hello()
hello

>>> def get_greetings(name):
>>>     return 'Hello, ' + name

>>> print get_greetings('Alex')
Hello, Alex
```

Немного о типах

```
>>> def empty():
>>>     return
```

- Возвращает специальный объект **None**

- Узнать тип объекта можно вызвав функцию **type(...)**:

```
>>> type('hello')
<type 'str'>
```

Финальные замечания

- Нет скобок – все регулируется отступами (4 пробела, см. policy)
- Оператор **for** для итерирования – аналог ***foreach***
- Стандартная библиотека содержит много полезного:

  https://docs.python.org/2.7/library | https://docs.python.org/3/library

Полезные ссылки

- Сайт Jupyter (инструкция по установке и документация – https://jupyter.org
- Как использовать сразу две версии Python: здесь и здесь
- Документация по «магическим» выражениям Jupyter – здесь